Automatic particle picking using diffusion filtering and random forest classification

Paul Joubert[1], Stephan Nickell[3], Florian Beck[3], Michael Habeck[1,2], Michael Hirsch[1], Bernhard Schölkopf[1]

[1]Max-Planck Institute for Intelligent Systems, Tübingen, Germany
[2]Max-Planck Institute for Developmental Biology, Tübingen, Germany
[3]Molecular Structural Biology, Max Planck Institute of Biochemistry, 82152 Martinsried, Germany

***Abstract*— An automatic particle picking algorithm for processing electron micrographs of a large molecular complex, the 26S proteasome, is described. The algorithm makes use of a coherence enhancing diffusion filter to denoise the data, and a random forest classifier for removing false positives. It does not make use of a 3D reference model, but uses a training set of manually picked particles instead. False positive and false negative rates of around 25% to 30% are achieved on a testing set. The algorithm was developed for a specific particle, but contains steps that should be useful for developing automatic picking algorithms for other particles.**

I. Introduction

Single particle cryo-electron microscopy (cryo-EM) is widely used in structural biology to elucidate three-dimensional structure information of macro-molecular assemblies (see [1] for an introduction to the field). By superimposing tens of thousands to hundreds of thousands of individual particle images, a reconstruction with sub-nanometer can be achieved. Acquisition of these large data sets can done by using automated routines while the extraction of the particles from the electron micrographs is often carried out by a manual, interactive procedure. There have been many attempts at automating this step (see [2] for a comparison between several different methods), but a general solution has yet to emerge. Current manual or semi-automatic approaches are very time-intensive, requiring several man-months for larger datasets, and thus forming a bottle-neck to obtaining higher resolution reconstructions.

In the comparative study in [2], 10 different algorithms for automatic particle selection are described. They can be divided into two general types of algorithms. In the first type, particles in a micrograph are detected by matching them to a set of templates. This is known as template matching. The templates could be generated using a low-resolution 3D reference model, or by averaging manually picked particles. An example of a template matching algorithm is [3]. The second type of algorithms are feature-based ones. Such algorithms compute features of image patches with which the patch can be classified as containing a particle or not. An example of a feature-based algorithm is [4]. Some algorithms can be considered as combinations of these two types.

The wide variety of possible solutions (as for example in [2]) reflects the large variability between the different datasets obtained from different particles. A given approach usually makes dataset-specific assumptions that do not generalize to other datasets. For instance, the method might rely on the signal-to-noise ratio (SNR) being above a certain threshold, or it might assume that the particle is roughly rotationally symmetric.

In this paper we take the view that to obtain the best algorithm for a given dataset, it should be designed specifically for that dataset. We describe such an algorithm that was developed specifically for a dataset of the 26S proteasome ([5]).

The algorithm does not make use of an initial 3D reference model. Instead, prior knowledge about the particle is introduced via a training step, using a small set of manually picked micrographs. According to the above classification, it could be considered to be a feature-based algorithm, although the features are learned, rather than explicitly described.

The algorithm consists of several different steps, some of which have previously been used for particle picking (denoising through diffusion, [6]), while others were first introduced in other contexts (random forests, [7] and [8]). Even though it was not tested on datasets of other particles, at least some of the steps should be useful in constructing algorithms for other, similar datasets.

II. Method

The algorithm operates on each raw micrograph, and proceeds through the following steps to produce the picks[1] for that micrograph:

- Segmentation
- Normalization
- Denoising
- Picking
- Alignment
- Classification

Each of these steps is described below in more detail.

A. Segmentation

The particles of interest lie inside a thin ice layer that is suspended within a carbon grid. Most micrographs contain

[1]A note on terminology: by a 'pick' we mean the (x, y)-coordinate of the center of a particle on a micrograph. The procedure of 'picking a micrograph' means generating picks for a given micrograph. This could be done either manually or automatically, resulting in either 'manual picks' or 'auto picks'.

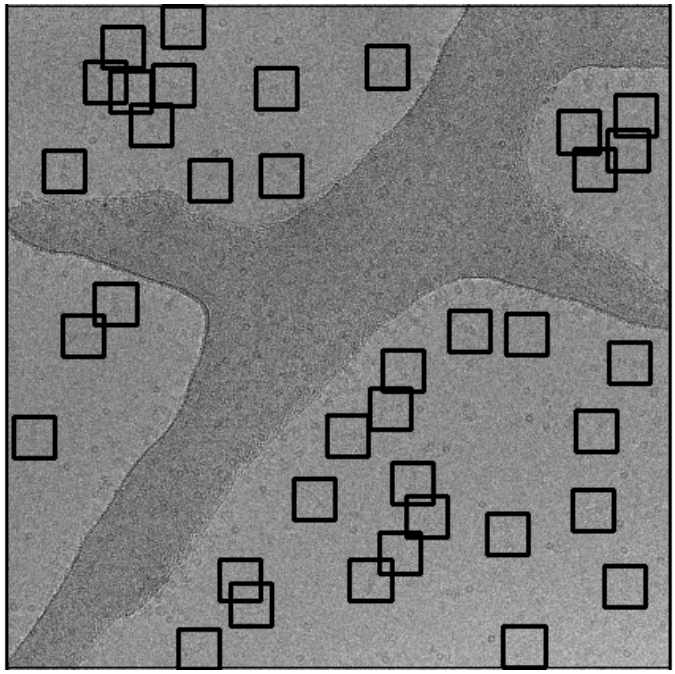


Fig. 1. An example of a raw transmission electron micrograph with a connected carbon region running diagonally and horizontally across the image. The manually picked particles (surrounded by black squares) are all outside the carbon strip. Note that the low signal-to-noise ratio makes it hard to see the particles.

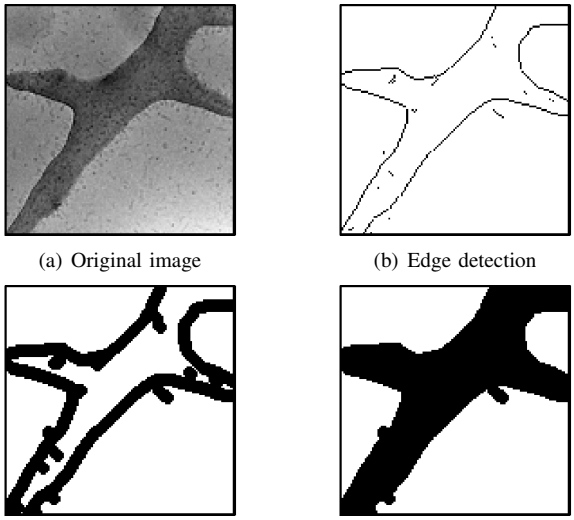
part of the carbon grid as well. See Figure 1 for an example. There are also particles on top of the carbon grid, but they tend to overlap, and are never manually picked. To ensure that no auto picks are placed here either, a mask is created in this first step to remove the carbon grid.

Figure 2 shows the steps used to generate such a mask. The first step is to perform edge detection, for which the approach introduced in [9] is used. The resulting edge map is thresholded to produce a binary edge map (see Figure 2(b)). The edges are then dilated with a disk to ensure that there are no gaps (Figure 2(c)). Each connected component of the part without edges is considered as a possible desired area (i.e. an area where particles should be picked). For example, in Figure 2(c) there are 5 such connected components. But the one in the center corresponds to a carbon strip, and should therefore be removed. This is done by calculating the convexity ratio

$$\eta = \frac{\text{area}(C)}{\text{area}(\overline{C})} \tag{1}$$

for each component C, where $\overline{C}$ denotes the convex hull of C, that is, the smallest convex set containing C. Components for which η is below a fixed threshold are removed, and the final mask is formed using the remaining components (Figure 2(d)).

This step requires a number of parameters to be set, such as the threshold for the edge map, the radius of the dilation disk and the convexity ratio. The final mask is not very sensitive to these parameters, and therefore they were just set manually. The same set of parameter values works for all micrographs.



(a) Original image (b) Edge detection

(c) Thicker edges (d) Final mask

Fig. 2. The steps involved in creating a mask. The original micrograph is downsampled to increase the signal-to-noise ratio (2(a)). Edges are detected using a pair of oriented energy filters (2(b)). The edges are thickened (2(c)) and connected components corresponding to carbon strips are detected and removed (2(d)).

B. Normalization

The thickness of the ice layer varies across the micrograph, and between micrographs. As a result, the local mean value of the recorded intensity varies gradually across the image. For example, in Figure 2(a) the image becomes lighter towards the bottom-right corner. For the subsequent denoising algorithm it is necessary that the image noise has roughly constant statistical properties across the image, and between different images. The purpose of this step is to normalize the micrograph in such a way that for any patch with size of the order of a particle, the mean and variance of the intensity values will be roughly 0 and 1 respectively.

The micrograph I is normalised by using a low-pass filter to estimate the local mean, and dividing by this mean field. That is,

$$I' = \frac{I}{I * f_\sigma} \tag{2}$$

where f_σ is a Gaussian with standard deviation σ and $*$ denotes the convolution operator. The resulting image is then shifted and scaled to have the desired mean and variance values:

$$I'' = \frac{I' - \text{mean}(I')}{\text{std}(I')}. \tag{3}$$

Note that for the raw image, the variation in mean value across the image is much smaller than the mean value itself, hence we are not introducing significant spatial correlations between the new variance and the original mean value.

The latter method can easily be adapted to take the mask into account. The resulting image is normalised in the desired areas, and zero elsewhere. The denoising algorithm detailed below doesn't work if there are large areas with zero value. Therefore we fill these regions with i.i.d. Gaussian noise.

C. Denoising

The micrographs are denoised using coherence enhancing diffusion, as described in [10]. This is an extension of the diffusion filter introduced in [11]. There, the image $f(x)$ is considered as the initial condition of a continuous sequence of images $u(x,t)$:

$$u(x,0) = f(x). \tag{4}$$

The evolution of u is governed by the differential equation

$$\partial_t u = \operatorname{div}(g(|\nabla u|^2)\nabla u) \tag{5}$$

where one possible choice for g is given by

$$g(|\nabla u|^2) = \frac{1}{1 + |\nabla u|^2/\lambda^2}. \tag{6}$$

If g were just a constant function, the filter would be equivalent to convolving with a Gaussian (see [10] for details). The effect of g can be understood as decreasing the standard deviation of this Gaussian kernel in the vicinity of strong edges, thereby preserving such edges.

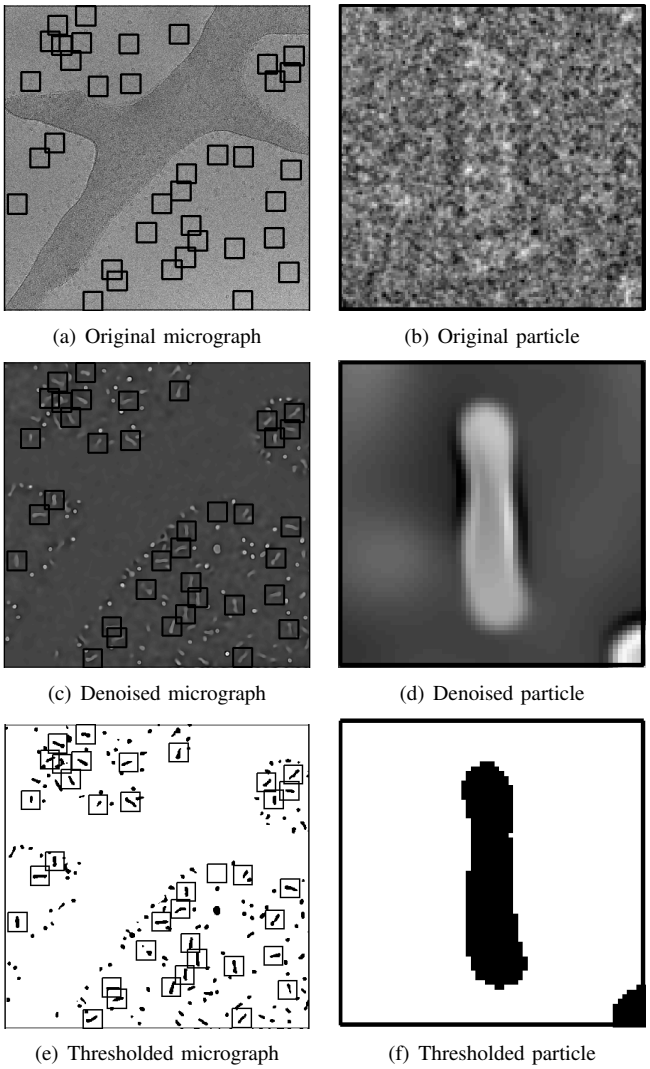
One shortcoming of the method is that the noise around edges is not diffused. To address this, the scalar g is replaced by a 2×2-matrix D called the diffusion tensor, making the filter anisotropic:

$$\partial_t u = \operatorname{div}(D \cdot \nabla u) \tag{7}$$

The matrix D is chosen so that diffusion is enhanced parallel to strong edges, but suppressed perpendicular to them. See [10] for details.

Figure 3 shows the effect of applying the filter on a micrograph and on an individual particle.

A very similar denoising algorithm was previously used for denoising cryo-EM images in [6].



(a) Original micrograph (b) Original particle

(c) Denoised micrograph (d) Denoised particle

(e) Thresholded micrograph (f) Thresholded particle

Fig. 3. Example of entire micrograph (on the left) and individual particle (on the right) during different steps of the algorithm. The first row shows the original micrograph and particle. The second row shows the effect of applying coherence enhancing diffusion to the images in the first row. The final row shows the result of thresholding the images from the second row.

D. Picking

In the denoised images, the particles are clearly visible (see Figure 3). A binary image is created by thresholding the denoised image with a fixed threshold value of 0.25. The connected components are then identified as being potential particles. Those connected components lying near the edge of the micrograph are removed. For each connected component, the pick is placed at the centroid of the component. This turns out to be very close to where the manual pick is placed in most cases.

The above approach yields a large number of false positives. Most of these false positives have a round shape, and correspond to proteasomes seen from the top, along their main axis. Hence they are referred to as top-views. Some of them are complete particles, but others are partially assembled proteasomes, such as ones with either one or both caps missing. They are usually not picked manually, and picking results are greatly improved by identifying them and removing them from the list of possible picks.

To decrease the number of false positives, a template-matching approach is used to identify the top-views in the denoised image so that they can be removed from the connected components afterwards. In particular, the top-views from a single denoised micrograph were picked (in an automatic fashion) and averaged to form a single template. This template is never changed. For subsequent denoised images, the template is cross-correlated with the image, and all peaks above a fixed threshold are annotated as top-views. Figure 4 shows an example of a micrograph and the top-views that were identified in this way.

The purpose of this step is to generate a set of picks that includes almost all the particles, but perhaps several non-particles as well. The undesired particles can then be removed in subsequent steps. In our experiments, around 90% of all true particles were usually included among the particles picked in this step. This means that 10% of the true particles will certainly not be part of the final set of picks. On the other

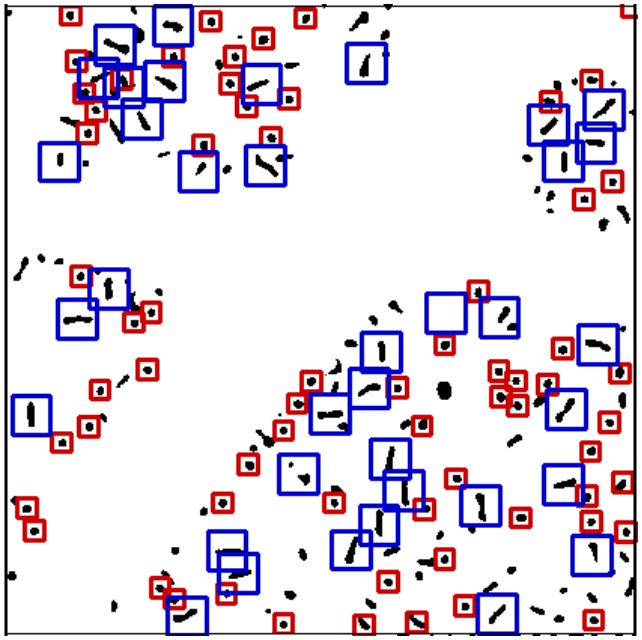


Fig. 4. Picking and removing top views. The components inside red squares were identified as top-views, while the blue squares indicate the positions of manually picked particles. The components that are left after removing the top-views will be picked automaticallly. Note that the auto picks include almost all the manual picks, and that many top-views are removed, none of which were manually picked.

hand, the number of false positives was typically about 2 to 3 times higher than the number of true particles. Most of them are removed in the later classification step.

E. Alignment

1) Motivation: After the picking step, we have a list of picks that contain many false positives. In the next step we train a classifier to get rid of them. The classifier is trained on a small set of picks from micrographs for which we have manual picks available.

A classifier requires some way of comparing picks, for instance by defining a way of computing distances between them. A simple way to do this, is to cut out a patch corresponding to the size of a particle around each pick, and computing the Euclidean norm between such patches.

In our case, the patches should not be cut out from the denoised images, but rather from the original micrographs. This is because some information is lost during denoising: a true particle and a false particle might look the same after being denoised, and thus be indistinguishable to a classifier.

The patches that are cut out from the original micrographs are normalised to account for variations in the mean and variance that are due to noise. After normalization, the distances between given pairs of patches are rather similar. Only when one patch has been shifted and rotated so that its particle is aligned to the particle contained in the other patch, does the Euclidean distance decrease measurably. Therefore there is an alignment step needed before classification to simplify the job of the classifier.

2) Alignment procedure: The alignment step also makes use of a training set. This would typically be the same (or a subset of the) set used for training the classifier. The picked particles from this set that correspond to manually picked particles are then aligned using a reference-based alignment procedure. During each step, each particle is aligned to the reference, and at the end of the step the reference is replaced by the average of all the aligned particles. Repeating this a few times yields a single template representing a typical particle.

This template is then used to align the training set. Each particle in the training set is aligned individually to the template, and the template is not changed in the process.

F. Classification

In this step a classifier is trained using a small number of training samples, and then applied to the aligned particles in order to remove as many false positives as possible.

A random forest with the binary features introduced in [7] is used as a classifier. Each such binary feature is described by an ordered pair of rectangles inside the image patch, R_1 and R_2, where each rectangle

$$R_i = \{c_{x1}, c_{y1}, c_{x2}, c_{y2}\} \tag{8}$$

requires 4 integers to determine its upper-left and lower-right corners. The feature is evaluated on an image patch

$$s = \{x_1, x_2, \ldots, x_n\} \tag{9}$$

by comparing the average intensity under the two rectangles:

$$f(s, R_1, R_2) = \Theta\left(\frac{\sum_{i|x_i \in R_1} x_i}{\sum_{i|x_i \in R_1} 1} - \frac{\sum_{i|x_i \in R_2} x_i}{\sum_{i|x_i \in R_2} 1}\right) \tag{10}$$

where $\Theta(\cdot)$ is true for positive values, and false otherwise. In other words, the binary feature returns true (or positive) if the average intensity under the first rectangle is greater than that in the second rectangle, and false (negative) otherwise.

A binary random forest classifier consists of a collection of decision trees, each of which is a binary classifier in its own right. Each decision tree has a binary feature as described above at each internal node, and a class assignment (positive or negative) attached to each leaf. A new patch is classified by starting at the root node, and recursively applying the binary feature at the current node to decide which branch to follow. When a leaf is reached, the class assignment of that leaf is the output of the given decision tree on the patch. The decisions of all the trees are combined using a majority vote.

Figure 5 shows an example of a binary feature. The feature is determined by the red and blue rectangles. It will tend to return true for horizontally aligned particles, since the expected average intensity within the first (red) rectangle is positive due to the presence of the particle, while the expected average intensity within the second (blue) rectangle is zero (due to normalisation). For non-particles, it might return true only half the time. This means that it has some ability to discriminate between true particles and non-particles. A decision tree might have such a feature at its root, with subsequent features

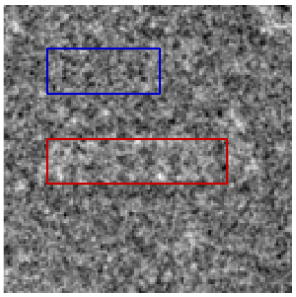


Fig. 5. An example of a binary feature applied to a particle. The feature is described by the two rectangles. It returns true if the mean value inside the red rectangle is higher than the mean value inside the blue rectangle. For this specific particle it returns true, because particles tend to have higher intensity values than their surroundings.

TABLE I

NUMBER OF PICKS IN TRAINING AND TESTING SETS.

	Manual picks	**Auto picks before classification**	**Auto picks after classification**
Training set	968	2242	-
Testing set	9863	25728	9276

focusing on smaller differences between particles and non-particles.

A decision tree is trained by randomly generating 1000 possible features at each node, and then selecting the one that is most able to discriminate between true and false particles. For more details, see [7].

III. RESULTS

Cryo-electron micrographs of the 26S proteasome recorded at the Department of Molecular Structural Biology, MPI of Biochemistry, Martinsried ([5]) were used to test the algorithm. A set of 220 micrographs was used. The first 20 micrographs were used for training the classifier, and the last 200 for testing. Table I shows the number of particles in the training and testing set. The number of auto picks before classification refers to the number of picks obtained by the algorithm after the picking step and after the top views have been removed, but before the particles have been aligned. The number of auto picks after classification is the final number of picks produced by the algorithm. From Table I it can be seen that before classification there are many more auto picks (25728) than true (manual) picks, but that the final number of picks (9276) is only slightly less than the number of particles picked manually (9863).

TABLE II

FALSE POSITIVE AND FALSE NEGATIVE RATES BEFORE AND AFTER CLASSIFICATION.

	Before classification	**After classification**
FPR	65.8%	26.0%
FNR	10.8%	30.4%

Table II summarizes the accuracy of the algorithm. If p_m and p_a denote the number of particles picked manually and automatically respectively, and p_{ma} is the number of particles picked both manually and automatically, then the false positive rate (FPR) and false negative rate (FNR) are defined as

$$\text{FPR} = \frac{p_a - p_{ma}}{p_a} \tag{11}$$

and

$$\text{FNR} = \frac{p_m - p_{ma}}{p_m}. \tag{12}$$

The overall performance of the algorithm is described by the final column. This shows that roughly one out of every four picks produced by the algorithm was not picked manually, and that about one third of all manual picks are missed by the algorithm. Note that a smaller false positive rate should result in a more accurate reconstruction, while a smaller false negative rate means that more micrographs need to be processed.

IV. DISCUSSION AND CONCLUSIONS

We described an algorithm for automatically picking particles of the 26S proteasome from cryo-electron micrographs. The algorithm achieves false positive and false negative rates of 26.0% and 30.4% respectively on the test set. It's difficult to put these results into perspective, and to decide what is good enough. One criterion could be to compare the picking results of different human pickers on the same dataset to each other. This was done in [2], yielding false positive and negative rates of 11.7% and 2.3% respectively. However, in [2] it is noted that the KLH dataset used in that comparative study represents a relatively easy dataset in particle selection. A similar comparison for the particle considered in this paper suggests that false positive/negative rates in the range 20% to 30% would be reasonable (results not shown).

One possible advantage of the proposed algorithm is that the picking step produces a set of all possibly interesting particles. Among these are incompletely assembled proteasomes. Such incomplete particles are not manually picked, and therefore discarded by the subsequent classification step. But should reconstruction algorithms improve to the point where they can make use of such incomplete particles as well, then they could make use of this intermediate output of the algorithm.

Several steps of the algorithm require a few thresholds to be set. They were all set manually, but then fixed for all the data. The parameters were not optimised to achieve the best results, as the results are not very sensitive to any of them. For applying the same algorithm to micrographs of the same particle generated by another electron microscope, it is likely that some thresholds would need adjustment.

For creating an autopicking algorithm for another particle, especially the denoising and classification steps of this algorithm might prove useful. Many other denoising methods were tested and found not to perform very well due to the low signal-to-noise ratio. Non-linear diffusion, and coherence enhancing diffusion in particular appear to be one of the few approaches that yielded good results.

The random forest classifier is also well-suited to this application. Its features involve the sum of intensities over (usually large) rectangles, canceling out the noise, and thereby making the features more insensitive to noise. The proper normalisation of particles is a complicated matter (see [12] or [13]), but this classifier is insensitive to normalisation as long as the scaling factor is positive. It is also fast and easy to train, and very fast to evaluate.

REFERENCES

[1] J. Frank, *Three-dimensional electron microscopy of macromolecular assemblies: visualization of biological molecules in their native state*, Oxford University Press, 2006.

[2] Y. Zhu, B. Carragher, R. M. Glaeser, D. Fellmann, C. Bajaj, M. Bern, F. Mouche, F. de Haas, R. J. Hall, D. J. Kriegman, S. J. Ludtke, S. P. Mallick, P. A. Penczek, A. M. Roseman, F. J. Sigworth, N. Volkmann, and C. S. Potter, "Automatic particle selection: results of a comparative study," *J. Struct. Biol.*, vol. 145, pp. 3–14, 2004.

[3] Z Huang and P Penczek, "Application of template matching technique to particle detection in electron micrographs," *Journal of Structural Biology*, vol. 145, no. 1-2, pp. 29–40, Jan. 2004.

[4] Yuanxin Zhu, Bridget Carragher, Fabrice Mouche, and Clinton S Potter, "Automatic particle detection through efficient Hough transforms.," *IEEE transactions on medical imaging*, vol. 22, no. 9, pp. 1053–62, Sept. 2003.

[5] S. Bohn, F. Beck, E. Sakata, T. Walzthoeni, M. Beck, R. Aebersold, F. Forster, W. Baumeister, and S. Nickell, "Structure of the 26S proteasome from Schizosaccharomyces pombe at subnanometer resolution," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 107, pp. 20992–20997, Dec 2010.

[6] Umesh Adiga, William T Baxter, Richard J Hall, Beate Rockel, Bimal K Rath, Joachim Frank, and Robert Glaeser, "Particle picking by segmentation: a comparative study with SPIDER-based manual particle picking.," *Journal of structural biology*, vol. 152, no. 3, pp. 211–20, Dec. 2005.

[7] T.J. Fuchs and J.M. Buhmann, "Inter-active learning of randomized tree ensembles for object detection," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. 2010, pp. 1370–1377, IEEE.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi, "Contour and texture analysis for image segmentation," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[10] Joachim Weickert, "Coherence-enhancing diffusion filtering," *International Journal of Computer Vision*, vol. 31, no. 2, pp. 111–127, 1999.

[11] Pietro Perona and Jitendra Malik, "Scale-space and edge detection using anisotropic diffusion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 7, pp. 629–639, 1990.

[12] C O S Sorzano, L G de la Fraga, R Clackdoyle, and J M Carazo, "Normalizing projection images: a study of image normalizing procedures for single particle three-dimensional electron microscopy.," *Ultramicroscopy*, vol. 101, no. 2-4, pp. 129–38, Nov. 2004.

[13] Sjors H W Scheres, Mikel Valle, Patricia Grob, Eva Nogales, and José-María Carazo, "Maximum likelihood refinement of electron microscopy data with normalization errors.," *Journal of structural biology*, vol. 166, no. 2, pp. 234–40, May 2009.